

Approximate Fitting of Circular Arcs when Two Points are Known

Alexander Gribov
Environmental Systems Research Institute
380 New York Street
Redlands, CA 92373
E-mail: agribov@esri.com

Abstract—The task of approximation of points with circular arcs is performed in many applications, such as polyline compression, noise filtering, and feature recognition. However, development of algorithms that perform a significant amount of circular arcs fitting require an efficient way of fitting circular arcs with complexity $O(1)$. The elegant solution to this task based on an eigenvector problem for a square nonsymmetrical matrix is described in [1]. For the compression algorithm described in [2], it is necessary to solve this task when two points on the arc are known. This paper describes a different approach to efficiently fitting the arcs and solves the task when one or two points are known.

Index Terms—arc fitting; optimization; compression; generalization

I. INTRODUCTION

The purpose of this paper is to solve the task of fitting a circular arc to a set of points (or segments) with complexity $O(1)$ when two points on the arc and moments up to the fourth order are known.

In papers [3] and [4], fitting a circle is done by finding such circle with center (x_c, y_c) and radius r , which minimizes the next equation

$$\sum_{i=1}^n \left(\left((x_i - x_c)^2 + (y_i - y_c)^2 \right) - r^2 \right)^2, \quad (1)$$

where (x_i, y_i) are i -th point, $i = \overline{1..n}$.

This formula is minimizing square differences between square distances from the circle center to the points and square of the radius. The solution is found by using only the moments of (x_i, y_i) with complexity $O(1)$. However, this leads to bias in the estimation of parameters [3, see pp. 368-370]. Suppose that each point has been fitted with ϵ_i error. Substituting it in (1) gives

$$\sum_{i=1}^n \left((r + \epsilon_i)^2 - r^2 \right)^2.$$

After simplifying this equation

$$\sum_{i=1}^n (2r \cdot \epsilon_i + \epsilon_i^2)^2 = \sum_{i=1}^n \left(\epsilon_i^2 (2r + \epsilon_i)^2 \right).$$

Assuming that ϵ_i are small compared to radius r and neglecting higher orders

$$4r^2 \sum_{i=1}^n \epsilon_i^2. \quad (2)$$

From this formula, it is clear that the fitting is trying to decrease the radius to minimize the equation (1). When the points cover only a small part of a circle, the estimated center of the circle can move toward the arc to reduce the radius. Despite that, the errors are increased, while overall penalty (1) is decreased. The smaller the angle of the arc, the worse the effect. Fig. 1 shows the case when fitting by moments produces a circle with too small a radius.

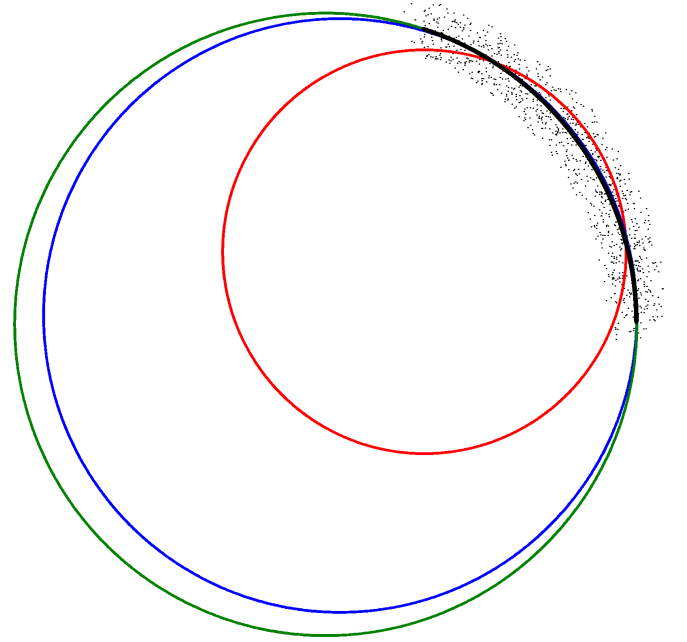


Fig. 1. Comparison of fitting an arc using different approaches. The comparison is performed for the arc with 72° , and uniform noise in the circle is proportional to 10 percent of the arc radius. A total of 1,000 random points were simulated along the arc with uniform steps. The black arc is a ground truth arc. The black dots are source points. The red circle is a solution based on fitting squares of distances (see (1)). The green circle is solution based on fitting distances (see (3)). The blue circle is a solution described in this paper (approximate solution of (5) found by one iteration of the algorithm described in Appendix B: Minimization of Multidimensional Function $f(x)$, $x \in \mathbb{R}^n$).

Minimization of the next equation was suggested in [5]:

$$\sum_{i=1}^n \left(\sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} - r \right)^2. \quad (3)$$

It has this error formula:

$$\sum_{i=1}^n \epsilon_i^2. \quad (4)$$

This doesn't have a problem like in (2). However, minimizing (3) requires an iterative approach, which analyzes all points (x_i, y_i) leading to an algorithm with complexity $O(n)$. The efficient algorithm to find the minimum of (3) is described in [6].

II. ALGORITHM

The solution to remove the square root from (3) was developed in [7], [8, see p. 675], [9], and [1]. From (2), there comes an idea that dividing (1) by $4r^2$ and minimizing it will produce a result closer to (3) because it is close to (4). The approximation based on the Taylor expansion of the square root by the first two terms gives exactly this solution. Approximation of \sqrt{x} at $x = 1$:

$$\sqrt{x} \approx 1 + \frac{1}{2}(x - 1) + O(x^2) = \frac{1}{2} + \frac{x}{2} + O(x^2).$$

Applying this approximation to (3)

$$\begin{aligned} \sum_{i=1}^n \left(\sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} - r \right)^2 &= \\ &= r^2 \sum_{i=1}^n \left(\sqrt{\frac{(x_i - x_c)^2 + (y_i - y_c)^2}{r^2}} - 1 \right)^2 \approx \\ &\approx r^2 \sum_{i=1}^n \left(\left(\frac{1}{2} + \frac{(x_i - x_c)^2 + (y_i - y_c)^2}{2r^2} \right) - 1 \right)^2 = \\ &= r^2 \sum_{i=1}^n \left(\frac{((x_i - x_c)^2 + (y_i - y_c)^2) - r^2}{2r^2} \right)^2 = \\ &= \frac{\sum_{i=1}^n \left(((x_i - x_c)^2 + (y_i - y_c)^2) - r^2 \right)^2}{4r^2}. \quad (5) \end{aligned}$$

Unlike (3), this formula can be minimized using only moments. The direct solution, based on conformal geometric algebra, is described in [1]. The solution is based on finding eigenvalues of a square nonsymmetric matrix [1, see (24)]. This can be done by Schur factorization. The eigenvector corresponding to the minimal non-negative eigenvalue is the solution. Care needs to be taken for the cases when there is a solution with close to zero eigenvalue due to round-off error.

In this paper, another approach to minimization of (5) will be considered.

III. MINIMIZATION OF (5)

Suppose we have a good estimate (x_e, y_e, r_e) . Let's optimize it in the next form:

$$(x_e + \Delta x, y_e + \Delta y, \sqrt{r_e^2 + \Delta x^2 + \Delta y^2 + \Delta r}).$$

This covers all possible values of (x_c, y_c, r) and gives a significant advantage for finding the minimum by removing the third and fourth order variables in the numerator of (5):

$$\begin{aligned} f(\Delta x, \Delta y, \Delta r) &= \\ &= \frac{\sum_{i=1}^n \left(\begin{aligned} &x_i^2 + y_i^2 - \\ &-2(x_e \cdot x_i + y_e \cdot y_i) + \\ &+ (x_e^2 + y_e^2 - r_e^2) + \\ &+ 2\Delta x(x_e - x_i) + \\ &+ 2\Delta y(y_e - y_i) - \\ &- \Delta r \end{aligned} \right)^2}{4(r_e^2 + \Delta x^2 + \Delta y^2 + \Delta r)}. \end{aligned}$$

Writing it from moments

$$\begin{aligned} f(\Delta x, \Delta y, \Delta r) &= \\ &= \frac{v + v_x \cdot \Delta x + v_y \cdot \Delta y + v_r \cdot \Delta r + \\ &\quad + v_{x,x} \cdot \Delta x^2 + v_{y,y} \cdot \Delta y^2 + \Delta r^2 + \\ &\quad + v_{x,y} \cdot \Delta x \cdot \Delta y + v_{x,r} \cdot \Delta x \cdot \Delta r + v_{y,r} \cdot \Delta y \cdot \Delta r}{4(r_e^2 + \Delta x^2 + \Delta y^2 + \Delta r)}, \quad (6) \end{aligned}$$

where

$$\begin{aligned} v &= (M_{4,0} + 2M_{2,2} + M_{0,4}) - \\ &\quad - 4(M_{3,0} + M_{1,2})x_e - 4(M_{2,1} + M_{0,3})y_e + \\ &\quad + 8M_{1,1} \cdot x_e \cdot y_e + 2M_{2,0} \cdot z_x + 2M_{0,2} \cdot z_y - \\ &\quad - 4(M_{1,0} \cdot x_e + M_{0,1} \cdot y_e)z + z^2, \\ v_x &= 4(-(M_{3,0} + M_{1,2}) + (3M_{2,0} + M_{0,2})x_e + \\ &\quad + 2M_{1,1} \cdot y_e - 2M_{0,1} \cdot x_e \cdot y_e - M_{1,0} \cdot z_x + x_e \cdot z), \\ v_y &= 4(-(M_{2,1} + M_{0,3}) + (M_{2,0} + 3M_{0,2})y_e + \\ &\quad + 2M_{1,1} \cdot x_e - 2M_{1,0} \cdot x_e \cdot y_e - M_{0,1} \cdot z_y + y_e \cdot z), \\ v_r &= -2(M_{2,0} + M_{0,2} - 2(M_{1,0} \cdot x_e + M_{0,1} \cdot y_e) + z), \\ v_{x,x} &= 4(M_{2,0} - 2M_{1,0} \cdot x_e + x_e^2), \\ v_{y,y} &= 4(M_{0,2} - 2M_{0,1} \cdot y_e + y_e^2), \\ v_{x,y} &= 8(M_{1,1} - M_{0,1} \cdot x_e - M_{1,0} \cdot y_e + x_e \cdot y_e), \\ v_{x,r} &= 4(M_{1,0} - x_e), \\ v_{y,r} &= 4(M_{0,1} - y_e), \\ z &= x_e^2 + y_e^2 - r_e^2, \\ z_x &= 3x_e^2 + y_e^2 - r_e^2, \\ z_y &= x_e^2 + 3y_e^2 - r_e^2, \\ M_{g,h} &= \frac{1}{n} \sum_{i=1}^n (x_i^g \cdot y_i^h). \end{aligned}$$

Minimization of (6) can be done using the approach described in Appendix B: Minimization of Multidimensional Function $f(x)$, $x \in \mathbb{R}^n$. To use that approach, it is sufficient

to know the matrix of second derivatives up to the constant

$$\begin{pmatrix} \frac{\partial^2 f}{\partial^2 \Delta x} & \frac{\partial^2 f}{\partial \Delta x \partial \Delta y} & \frac{\partial^2 f}{\partial \Delta x \partial \Delta r} \\ \frac{\partial^2 f}{\partial \Delta x \partial \Delta y} & \frac{\partial^2 f}{\partial^2 \Delta y} & \frac{\partial^2 f}{\partial \Delta y \partial \Delta r} \\ \frac{\partial^2 f}{\partial \Delta x \partial \Delta r} & \frac{\partial^2 f}{\partial \Delta y \partial \Delta r} & \frac{\partial^2 f}{\partial^2 \Delta r} \end{pmatrix} \sim \begin{pmatrix} d_{x,x} & d_{x,y} & d_{x,r} \\ d_{x,y} & d_{y,y} & d_{y,r} \\ d_{x,r} & d_{y,r} & d_{r,r} \end{pmatrix},$$

where

$$\begin{aligned} d_{x,x} &= -2(v - v_{x,x} \cdot r_e^2) \cdot r_e^2, \\ d_{x,y} &= v_{x,y} \cdot r_e^4, \\ d_{y,y} &= -2(v - v_{y,y} \cdot r_e^2) \cdot r_e^2, \\ d_{x,r} &= (-v_x + v_{x,r} \cdot r_e^2) \cdot r_e^2, \\ d_{y,r} &= (-v_y + v_{y,r} \cdot r_e^2) \cdot r_e^2, \\ d_{r,r} &= 2(v - v_r \cdot r_e^2 + r_e^6). \end{aligned}$$

and the equation for directional search by direction $(\alpha_x, \alpha_y, \alpha_r)$ is

$$\frac{v + (v_x \cdot \alpha_x + v_y \cdot \alpha_y + v_r \cdot \alpha_r)t + (v_{x,x} \cdot \alpha_x^2 + v_{y,y} \cdot \alpha_y^2 + \alpha_r^2 + v_{x,y} \cdot \alpha_x \cdot \alpha_y + v_{x,r} \cdot \alpha_x \cdot \alpha_r + v_{y,r} \cdot \alpha_y \cdot \alpha_r)t^2}{4(r_e^2 + \alpha_r \cdot t + (\alpha_x^2 + \alpha_y^2)t^2)}.$$

Looking at the numerator of (5), it would be reasonable to take a solution of (1), described in [3], as a starting point.

Only a few iterations are needed to converge beyond machine precision. However, this approach is only approximate, and there is no need for such precision. In practice, one iteration is sufficient to get a good approximation.

When estimation of the center is known, the best estimation of the radius can be easily found. However, it does not give any improvement in speed.

An approximation of the sum of square deviations from the polyline to an arc with the center (x_e, y_e) and radius r_e (see (3)) is found from (6) by setting Δx , Δy , and Δr to zero and multiplying by n .

$$n \frac{v}{4r_e^2} \quad (7)$$

The algorithm described in [1] has an advantage for finding the global optimum, while the algorithm described in this paper can go to the local optimum. This is likely to happen when the arc is close to the line. Otherwise, the results are identical.

The advantage of the approach described in this paper is the ability to reduce the amount of calculation by approximating the solution.

I have implemented both approaches. Intel Math Kernel Library 11.2 was used to solve the nonsymmetric eigenvector problem in [1]. The approach described in this paper is several times faster. However, it is difficult to make a fair comparison due to the different ways of implementing and optimizing the

code. When speed is not a concern, the approach described in [1] is preferred.

IV. OPTIMAL ARC WHEN ONE POINT IS KNOWN

For some tasks, one point on the arc is known in advance. The arc should pass through that point. Knowing the position of the center determines the radius: $r = \sqrt{(x_c - x_a)^2 + (y_c - y_a)^2}$, where (x_a, y_a) is a point on the arc. Substituting this into (5)

$$\frac{\sum_{i=1}^n \left(\left((x_i - x_c)^2 + (y_i - y_c)^2 \right) - \left((x_c - x_a)^2 + (y_c - y_a)^2 \right) \right)^2}{4 \left((x_c - x_a)^2 + (y_c - y_a)^2 \right)}. \quad (8)$$

The solution described in section III: Minimization of (5) can be applied. The differences are that the optimization is performed in two-dimensional space and the initial solution can be found from the least squares approach.

Multiplying the numerator and denominator of (8) by s^2 and replacing $x_c \cdot s$ and $y_c \cdot s$ by v_x and v_y , respectively, and setting $v = \begin{pmatrix} v_x \\ v_y \\ s \end{pmatrix}$

$$\frac{v^\top A v}{4 v^\top B v}, \quad (9)$$

where

$$A = \begin{pmatrix} a_{x,x} & a_{x,y} & a_{x,1} \\ a_{x,y} & a_{y,y} & a_{y,1} \\ a_{x,1} & a_{y,1} & a_{1,1} \end{pmatrix},$$

$$\begin{aligned} a_{x,x} &= 4(M_{2,0} - 2M_{1,0} \cdot x_a + x_a^2), \\ a_{y,y} &= 4(M_{0,2} - 2M_{0,1} \cdot y_a + y_a^2), \\ a_{1,1} &= M_{4,0} + 2M_{2,2} + M_{0,4} - \\ &\quad - 2(M_{2,0} + M_{0,2})(x_a^2 + y_a^2) + \\ &\quad + (x_a^2 + y_a^2)^2, \\ a_{x,y} &= 4(M_{1,1} - M_{1,0} \cdot y_a - M_{0,1} \cdot x_a + x_a y_a), \\ a_{x,1} &= -2(M_{3,0} + M_{1,2} - (M_{2,0} + M_{0,2})x_a - \\ &\quad - M_{1,0}(x_a^2 + y_a^2) + (x_a^2 + y_a^2)x_a), \\ a_{y,1} &= -2(M_{2,1} + M_{0,3} - (M_{2,0} + M_{0,2})y_a - \\ &\quad - M_{0,1}(x_a^2 + y_a^2) + (x_a^2 + y_a^2)y_a), \\ B &= \begin{pmatrix} 1 & 0 & -x_a \\ 0 & 1 & -y_a \\ -x_a & -y_a & x_a^2 + y_a^2 \end{pmatrix}. \end{aligned}$$

(9) is a generalized Rayleigh quotient. Note that A and B are symmetric non-negative matrices. The solution can be found by the generalized singular value decomposition of \sqrt{A} and \sqrt{B} . Square root matrices can be found from singular value decompositions such as

$$\begin{aligned} \sqrt{A} &= L_A^{\frac{1}{2}} X_A^\top, \\ \sqrt{B} &= L_B^{\frac{1}{2}} X_B^\top, \end{aligned} \quad (10)$$

where L_A and L_B are eigenvalue matrices of A and B , respectively; X_A and X_B are eigenvector matrices of A and B , respectively.

From generalized singular value decomposition for \sqrt{A} and \sqrt{B} follows

$$\begin{aligned}\sqrt{A} &= U D_A(0, R) Q^\top, \\ \sqrt{B} &= V D_B(0, R) Q^\top,\end{aligned}\quad (11)$$

where U , V , and Q are orthogonal matrices; R is an upper triangular matrix.

From (10) and (11) follows

$$\begin{aligned}A &= Q(0, R)^\top D_A^2(0, R) Q^\top, \\ B &= Q(0, R)^\top D_B^2(0, R) Q^\top.\end{aligned}$$

The smallest ratio of squares of eigenvalues is the solution. The center of the arc is recovered from v as $\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \end{pmatrix} \cdot \begin{pmatrix} 1/s \\ 1/s \end{pmatrix}$.

V. OPTIMAL ARC WHEN TWO POINTS ARE KNOWN

There are tasks when the starting and ending points of the arc are known (or any two points lying on the arc). Therefore, the center of the arc should lie on some line of $(x_c, y_c) = (x_p + \alpha \cdot t, y_p + \beta \cdot t)$, where (x_p, y_p) is a point on the line, (α, β) is the direction of the line ($\alpha^2 + \beta^2 = 1$), and t is any value. Knowing the position of the center determines the radius: $r = \sqrt{((x_p + \alpha \cdot t) - x_a)^2 + ((y_p + \beta \cdot t) - y_a)^2}$, where (x_a, y_a) is one of the points on the arc. Substituting this in (5)

$$\sum_{i=1}^n \left(\frac{\left((x_i - (x_p + \alpha \cdot t))^2 + (y_i - (y_p + \beta \cdot t))^2 \right) - \left(((x_p + \alpha \cdot t) - x_a)^2 + ((y_p + \beta \cdot t) - y_a)^2 \right)}{4 \left(((x_p + \alpha \cdot t) - x_a)^2 + ((y_p + \beta \cdot t) - y_a)^2 \right)} \right)^2. \quad (12)$$

It is easy to see that the minimum of (12) can be easily found because it has the form of (13) in Appendix A: Finding the Global Minimum of the Ratio of Quadratic Equations. An iterative algorithm is not needed to solve this problem.

The algorithm described in [2] finds an optimal polyline within the tolerance of the source polyline, with the minimum number of vertices, and among them, with the minimum sum of the square deviations from the optimal polyline. Extending this algorithm to support arcs requires efficient fitting of the arc from the known start and end points and evaluation of the sum of the square deviations from the source polyline to an arc. An approximate solution (7) can be used instead of direct evaluation (3).

VI. EXAMPLE: RECOVERING ARCS IN A CADASTRAL DATASET

The approach described in this paper for efficiently fitting circular arcs is used in a compression algorithm, when vertices of the source polylines are not allowed to move. The algorithm minimizes the weighted number of segments (with penalty 2) and arcs (with penalty 3) while satisfying tolerance restrictions. Among all possible solutions, the solution with

the minimum sum of square deviations is chosen. A dynamic programming approach was used to find the optimal solution [2].

Algorithms with a very similar framework were described in [10], [11].

Fitting of arcs in [10] was performed by checking tolerance when starting and ending vertices are fixed. It has the advantage of always finding an arc within tolerance; however, the computational complexity for each fitting is $O(n)$. This paper uses approximation to least squares fitting with complexity $O(1)$ described in section V: Optimal Arc when Two Points are Known. Although checking for the tolerance and proper sequence (zigzag) [2], [6], [10] has complexity $O(n)$, it is only performed for optimal fits.

The penalty function in [11] is a combination of perceptual and fitting errors. The perceptual error is

$$\delta \cdot \sin \frac{\alpha}{2},$$

where δ is the segmentation penalty and α is the angle between adjacent segments. This gives preference to solutions with acute angles.

An example is shown in Fig. 2. The original arcs were lost due to digitalization, limitations of the format, projection, and so forth. The restoration of arcs is an important task because it is the original representation. Restoring original arcs creates cleaner databases and simplifies future editing.

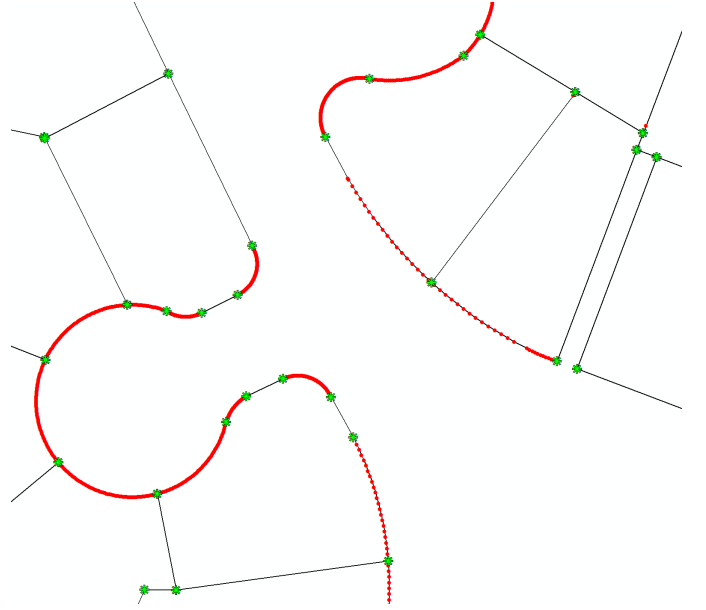


Fig. 2. Part of a parcel map with lost circular arcs. A compression algorithm was applied to this data. The Black lines are the source polylines, the red circles are vertices of the source polylines, and the green asterisks are resultant vertices. All original arcs were reconstructed.

The optimizations used for fitting segments are described in [2]. Initial fitting of circular arcs to different parts of the polyline was performed to reduce searching for arcs in a dynamic programming approach. The processing speed is about 0.03 seconds per 100 points on an Intel Xeon CPU E5-2670.

VII. FUTURE WORK

Described at the end of section VI: Example: Recovering Arcs in a Cadastral Dataset, reducing the search for arcs is only approximate and in some cases might skip possible solutions. There is an exact solution to check if an arc exists that covers all vertices within the specified tolerance described in [12, section 7.4]. The solution is found using the closest and the farthest point Voronoi diagrams. The center of an arc corresponding to the minimum width covering all vertices is either a vertex of the closest or the farthest point Voronoi diagram or a point on the edge of the closest and the farthest point Voronoi diagrams [12, p. 167]. The closest and the farthest point Voronoi diagrams are dual with the closest and the farthest point Delaunay triangulations, respectively [13]. Note that the farthest point Voronoi diagram includes only vertices on the convex hull. The algorithm to construct the farthest point Delaunay triangulation is similar to the algorithm for the closest point Delaunay triangulation and can be found in [14]. Another algorithm to construct the closest and the farthest point Delaunay triangulations for vertices on the convex hull is by mapping each vertex (x, y) to a vertex in three dimensional space $(x, y, x^2 + y^2)$ and constructing a convex hull for them [13].

VIII. CONCLUSION

This paper describes an efficient approximation for fitting circular arcs. While all formulas are for a two-dimensional case, the algorithm can be generalized for higher dimensions (for example, fitting a sphere to points).

The direct solution to fit arcs is described in [1]. This paper extends the solution to cases when one or two points on the arc are known.

Because the solution is based on fourth orders, it has a negative impact on the precision of calculations. This can be solved by shifting data to the origin of a coordinate system and/or using floating point numbers with a larger mantissa.

There is no evaluation of how well the fit is done. An additional algorithm is necessary to perform this check, as described in [6, see section 3].

APPENDIX A: FINDING THE GLOBAL MINIMUM OF THE RATIO OF QUADRATIC EQUATIONS

$$\frac{a_0 + a_1 \cdot x + a_2 \cdot x^2}{b_0 + b_1 \cdot x + b_2 \cdot x^2}, \quad (13)$$

where a_i and b_i are known coefficients $i = \overline{0..2}$.

Coefficients should satisfy

$$\forall x, a_0 + a_1 \cdot x + a_2 \cdot x^2 \geq 0. \quad (14)$$

Two cases will be analyzed separately:

1. $b_2 \neq 0$.

The domain will be restricted to

$$Q = \{b_0 + b_1 \cdot x + b_2 \cdot x^2 > 0\}. \quad (15)$$

Notice, limits (13) when $x \rightarrow -\infty$ and $x \rightarrow +\infty$ are the same.

The first derivative of (13) equals

$$\frac{(a_1 \cdot b_0 - a_0 \cdot b_1) + 2(a_2 \cdot b_0 - a_0 \cdot b_2) \cdot x + (a_2 \cdot b_1 - a_1 \cdot b_2) \cdot x^2}{(b_0 + b_1 \cdot x + b_2 \cdot x^2)^2}. \quad (16)$$

From (15), it follows that the denominator (16) is always positive in Q . Therefore, it is sufficient to work with the numerator:

$$c_0 + c_1 \cdot x + c_2 \cdot x^2, \quad (17)$$

where $c_0 = a_1 \cdot b_0 - a_0 \cdot b_1$, $c_1 = 2(a_2 \cdot b_0 - a_0 \cdot b_2)$, and $c_2 = a_2 \cdot b_1 - a_1 \cdot b_2$.

From (14), it follows that (13) is not negative in Q . If the denominator of (13) has real roots, then (13), when x is approaching any root, goes to $+\infty$ in Q and $-\infty$ in the complement of Q excluding roots (see example in Fig. 3). Local extrema are found from roots of (17) (Fig. 4). There is a special case, when in (13) the numerator is equal to zero at one of the roots of the denominator. In this case, (13) simplifies to the ratio of linear equations and doesn't have any global minimum.

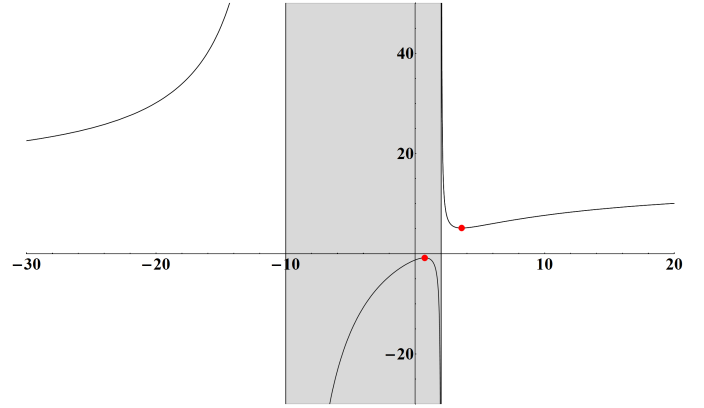


Fig. 3. Example of (13). The area outside domain Q is shown in gray. Local extrema are shown by red circles, found as the solution of (17) (see Fig. 4).

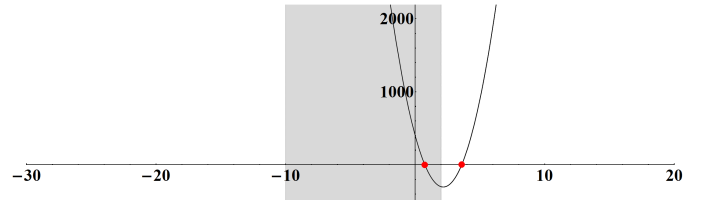


Fig. 4. Example of (17) corresponding to the function shown in Fig. 3. Roots are shown by red circles.

The global minimum can be found from roots of the quadratic equation (17):

- If $c_2 > 0$ and the largest root of (17) belongs to Q , then it is a global minimum.
- If $c_2 < 0$ and the smallest root of (17) belongs to Q , then it is a global minimum.

- c. If $c_2 = 0$, $c_1 > 0$ and the single root of (17) belongs to Q , then it is a global minimum.
- d. Otherwise, no global minimum exists.

To summarize, the global minimum of (13) can be found by the next equation if the value inside Q

$$\begin{cases} -\frac{c_0}{c_1} & \text{if } c_2 = 0 \wedge c_1 > 0, \\ \frac{\sqrt{D} - c_1}{2c_2} & \text{if } c_2 \neq 0 \wedge D > 0 \wedge c_1 < 0, \\ \text{sign}(c_2) \cdot \sqrt{-\frac{c_0}{c_2}} & \text{if } c_2 \neq 0 \wedge D > 0 \wedge c_1 = 0, \\ -\frac{2c_0}{\sqrt{D} + c_1} & \text{if } c_2 \neq 0 \wedge D > 0 \wedge c_1 > 0, \\ \text{no solution} & \text{otherwise,} \end{cases} \quad (18)$$

where $D = c_1^2 - 4c_0 \cdot c_2$ is discriminant of (17).

- 2. $b_2 = 0$. It is sufficient to evaluate the solution of the next equation to show that this case can be properly solved by 1:

$$\frac{a_0 + a_1 \cdot x + a_2 \cdot x^2}{x},$$

where a_i are known coefficients $i = \overline{0..2}$.

The domain will be restricted to

$$Q = \{x > 0\}.$$

The first derivative multiplied by x^2 equals

$$-a_0 + a_2 \cdot x^2.$$

From that global minimum

$$\begin{cases} \sqrt{\frac{a_0}{a_2}} & \text{if } a_0 \cdot a_2 > 0, \\ \text{no solution} & \text{otherwise,} \end{cases} \quad (19)$$

Notice that solution (19) is equal to solution (18). Therefore, it is sufficient to use (18) for both cases.

Another way to prove that the solution for the case 1 gives the proper solution (when $b_2 = 0$) is to consider $\lim_{b_2 \rightarrow 0}$ of (18).

APPENDIX B: MINIMIZATION OF MULTIDIMENSIONAL FUNCTION $f(x)$, $x \in \mathbb{R}^n$

Suppose that the minimum of $f(x)$ along any direction can be easily found as well as second derivatives.

The next algorithm is suggested:

- a. Let $i = 0$. Define the starting point x_0 .
- b. Find the second derivative matrix at x_i and find all eigenvectors.
- c. For each eigenvector, from x_i point, search along the eigenvector direction for minimum x_{i+1} . Set $i = i + 1$. Because the number of eigenvectors is n , this step increases the index of x by n .
- d. If x_i is close to the minimum with enough precision (for example, by comparing with the previous estimate x_{i-n}), then stop; otherwise, go to step b.

In the case of quadratic functions, this algorithm converges to the minimum in one iteration consisting of searching from any starting point by n direction.

REFERENCES

- [1] L. Dorst, "Total least squares fitting of k-spheres in n-d Euclidean space using an (n+2)-d isometric representation," *Journal of Mathematical Imaging and Vision*, vol. 50, no. 3, pp. 214–234, 2014. [Online]. Available: <http://doi.org/10.1007/s10851-014-0495-2>
- [2] A. Gribov, "Searching for a compressed polyline with a minimum number of vertices," *ArXiv e-prints*, April 2015. [Online]. Available: <http://arxiv.org/abs/1504.06584>
- [3] S. M. Thomas and Y. T. Chan, "A simple approach for the estimation of circular arc center and its radius," *Computer Vision, Graphics, and Image Processing*, vol. 45, no. 3, pp. 362–370, March 1989. [Online]. Available: [http://doi.org/10.1016/0734-189X\(89\)90088-1](http://doi.org/10.1016/0734-189X(89)90088-1)
- [4] C. Ichoku, B. Deffontaines, and J. Chorowicz, "Segmentation of digital plane curves: A dynamic focusing approach," *Pattern Recognition Letters*, vol. 17, no. 7, pp. 741–750, June 1996. [Online]. Available: [http://doi.org/10.1016/0167-8655\(96\)00015-3](http://doi.org/10.1016/0167-8655(96)00015-3)
- [5] S. M. Robinson, "Fitting spheres by the method of least squares," *Communications of the ACM*, vol. 4, no. 11, p. 491, November 1961. [Online]. Available: <http://doi.org/10.1145/366813.366824>
- [6] E. Bodansky and A. Gribov, "Approximation of polylines with circular arcs," in *Graphics Recognition. Recent Advances and Perspectives*, ser. Lecture Notes in Computer Science, J. Lladós and Y.-B. Kwon, Eds. Springer Berlin Heidelberg, 2004, vol. 3088, pp. 193–198. [Online]. Available: http://doi.org/10.1007/978-3-540-25977-0_18
- [7] V. Pratt, "Direct least-squares fitting of algebraic surfaces," *SIGGRAPH '87 Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, vol. 21, no. 4, pp. 145–152, Aug. 1987. [Online]. Available: <http://doi.org/10.1145/37402.37420>
- [8] G. Lukács, R. Martin, and D. Marshall, "Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation," in *Computer Vision - ECCV'98*, ser. Lecture Notes in Computer Science, H. Burkhardt and B. Neumann, Eds. Springer Berlin Heidelberg, 1998, vol. 1406, pp. 671–686. [Online]. Available: <http://doi.org/10.1007/BFb0055697>
- [9] A. Fitzgibbon, M. Pilu, and R. B. Fisher, "Direct least square fitting of ellipses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 476–480, May 1999. [Online]. Available: <http://doi.org/10.1109/34.765658>
- [10] A. Safonova and J. Rossignac, "Compressed piecewise-circular approximations of 3D curves," *Computer-Aided Design*, vol. 35, pp. 533–547, May 2003. [Online]. Available: [http://dx.doi.org/10.1016/S0010-4485\(02\)00073-8](http://dx.doi.org/10.1016/S0010-4485(02)00073-8)
- [11] L. Yin, Y. Yajie, and L. Wenyin, "Online segmentation of freehand stroke by dynamic programming," in *Eighth International Conference on Document Analysis and Recognition*, vol. 1, August 2005, pp. 197–201. [Online]. Available: <http://doi.org/10.1109/ICDAR.2005.180>
- [12] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd ed. Santa Clara, CA, USA: Springer-Verlag TELOS, 2008.
- [13] D. Eppstein, "The farthest point Delaunay triangulation minimizes angles," *Computational Geometry*, vol. 1, no. 3, pp. 143–148, 1992. [Online]. Available: [http://doi.org/10.1016/0925-7721\(92\)90013-I](http://doi.org/10.1016/0925-7721(92)90013-I)
- [14] K. Mehlhorn and S. Näher, *A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, June 2009.